EECS 370

Multi-Level Page Tables

Poll

What's your favorite Thanksgiving dish?

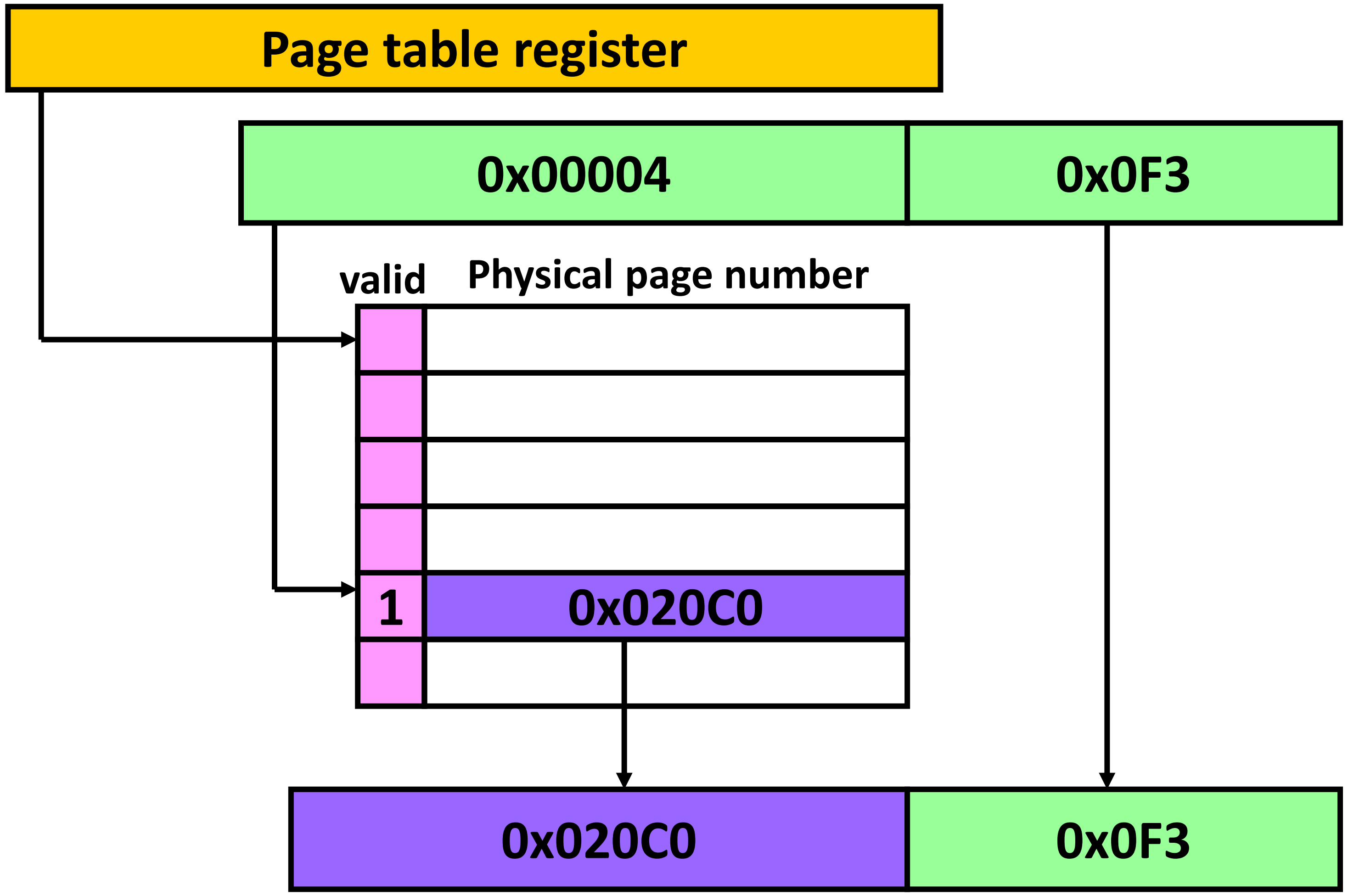
a) Turkey
b) Stuffing
c) Cranberry sauce
d) Sweet, delicious Buckeye tears

Announcements

- P4
  - Last project!
  - Due Thur (11/30)
- HW 4
  - Last homework!
  - Due Mon (12/4)
- Lecture Thursday
  - Last lecture!
  - Review on Tuesday
- Lab Fr/M
  - Last lab!
  - Don't forget to submit lab 11 by Wed
- Final exam
  - …Last exam!
  - Tue (12/12) @ 10:30 am

Reminder: Page tables

Virtual address = 0x000040F3



Physical address = 0x020C00F3

Agenda

- **Motivation for Multi-level Page Tables**
- Example architecture
- Class Problem: 32bit Intel x86
- Class Problem: Multi-Level VM Design
- VM Miscellanea

Size of the page table

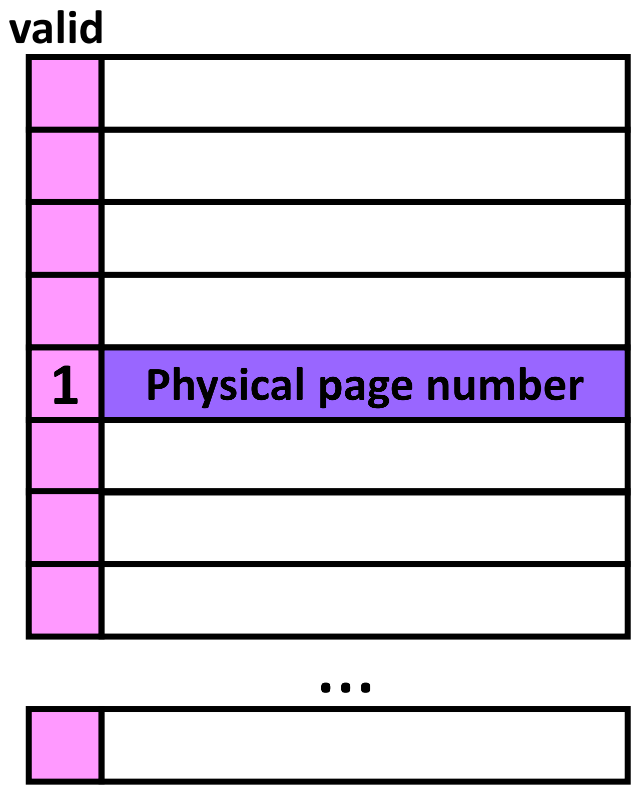
- How big is a page table entry?
  - For 32-bit virtual address:
    - If the machine can support 1GB = 2^{30} bytes of physical memory and we use pages of size 4KB = 2^{12},
    - then the physical page number is 30-12 = 18 bits.
      Plus another valid bit + other useful stuff (read only, dirty, etc.)
    - Let say about 3 bytes.
- How many entries in the page table?
  - 1 entry per virtual page
  - ARM virtual address is 32 bits – 12 bit page offset = 20
  - Total number of virtual pages = 2^{20}
- Total size of page table = Number of virtual pages

  * Size of each page table entry

  = $2^{20} \times 3$ bytes ~ 3 MB

How can you organize the page table?

1. Single-level page table occupies continuous region in physical memory
   - Previous example always takes 3MB regardless of how much virtual memory is used



A small program will use up 3MB to store just one PPN!!

It will be even worse for a 64-bit system!

How can you organize the page table?

2. Option 2: Use a multi-level page table
   - 1st level page table (much smaller!) holds addresses 2nd level page tables
     - 2nd level page tables hold translation info, or 3rd level page tables if we wanna go deeper
     - Only allocate space for 2nd level page tables that are used

valid	PPN
1	0x1
1	0x2
1	0x3

Single-level: Tons of wasted space!

valid	2nd level page table
1	0x1000

valid	PPN
1	0x1
1	0x2
1	0x3

Multi-Level Page Table

- Only allocate second (and later) page tables when needed
- Program starts: everything is invalid, only first level is allocated

Single Level

valid	2nd level page table
0	
0	
0	
0	

Multi-level: Size is proportional to amount of memory used

- As we access more, second level page tables are allocated

Multi Level

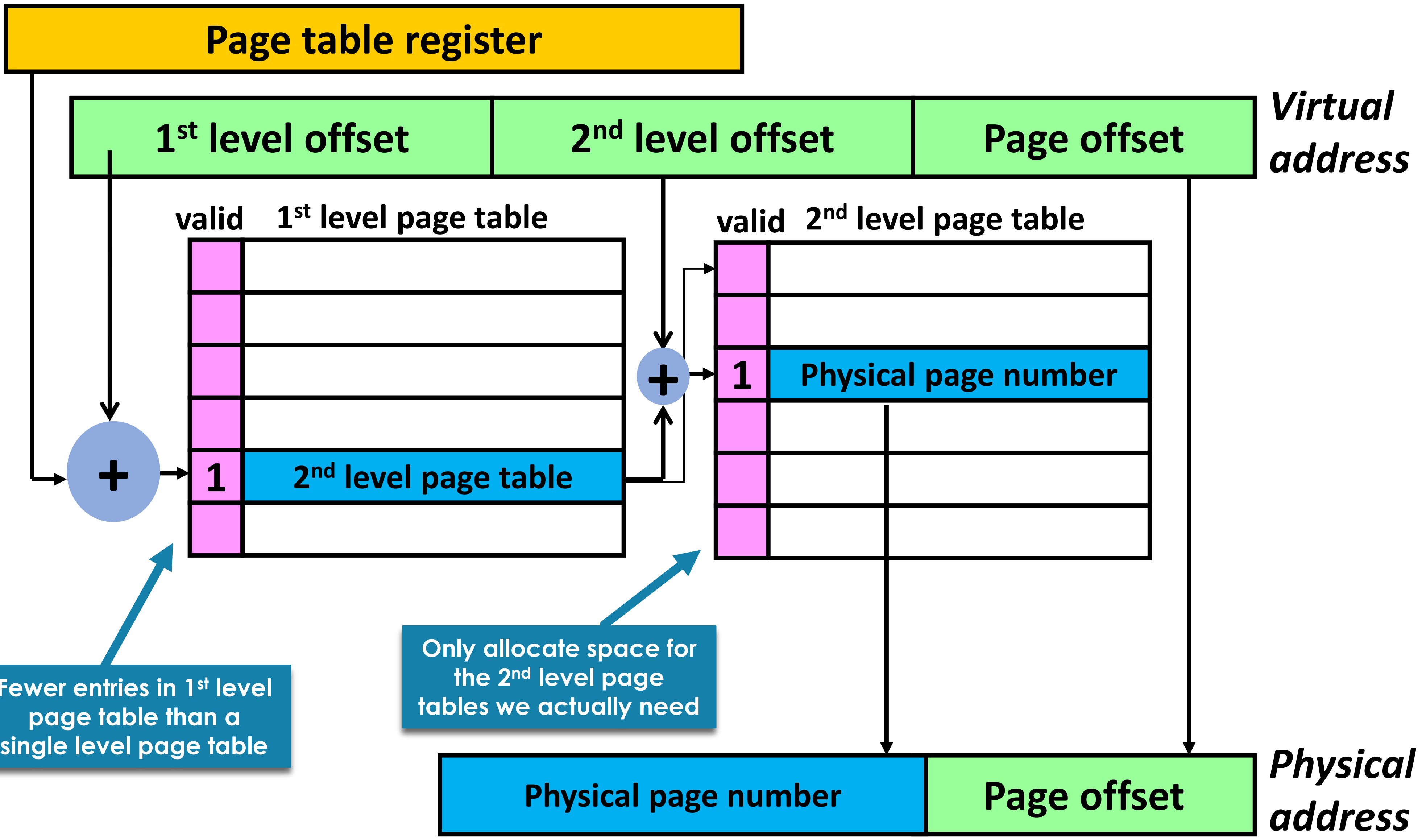
valid	2nd level page table
1	0x1000
1	0x3500

valid	PPN
1	0x1
1	0x6
1	0x2
1	0x1f

valid	PPN
1	0x9a
1	0x3
0	
1	0xff

Common case: most programs use small portion of virtual memory space

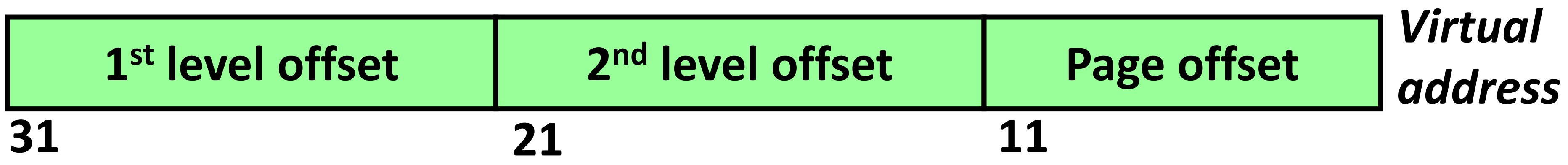
Hierarchical page table



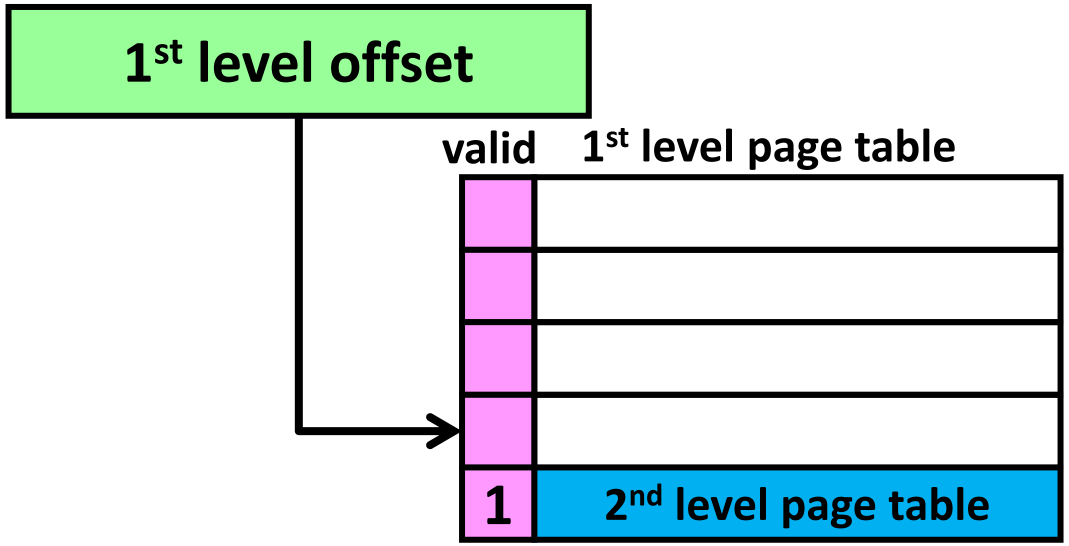
Agenda

- Motivation for Multi-level Page Tables
- **Example architecture**
- Class Problem: 32bit Intel x86
- Class Problem: Multi-Level VM Design
- VM Miscellanea

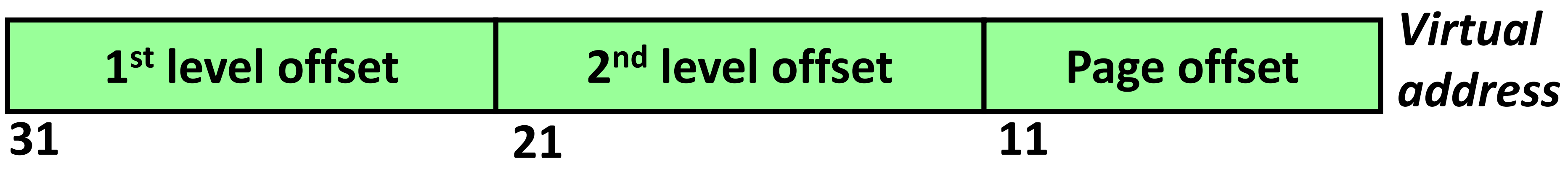
Hierarchical page table – 32bit Intel x86



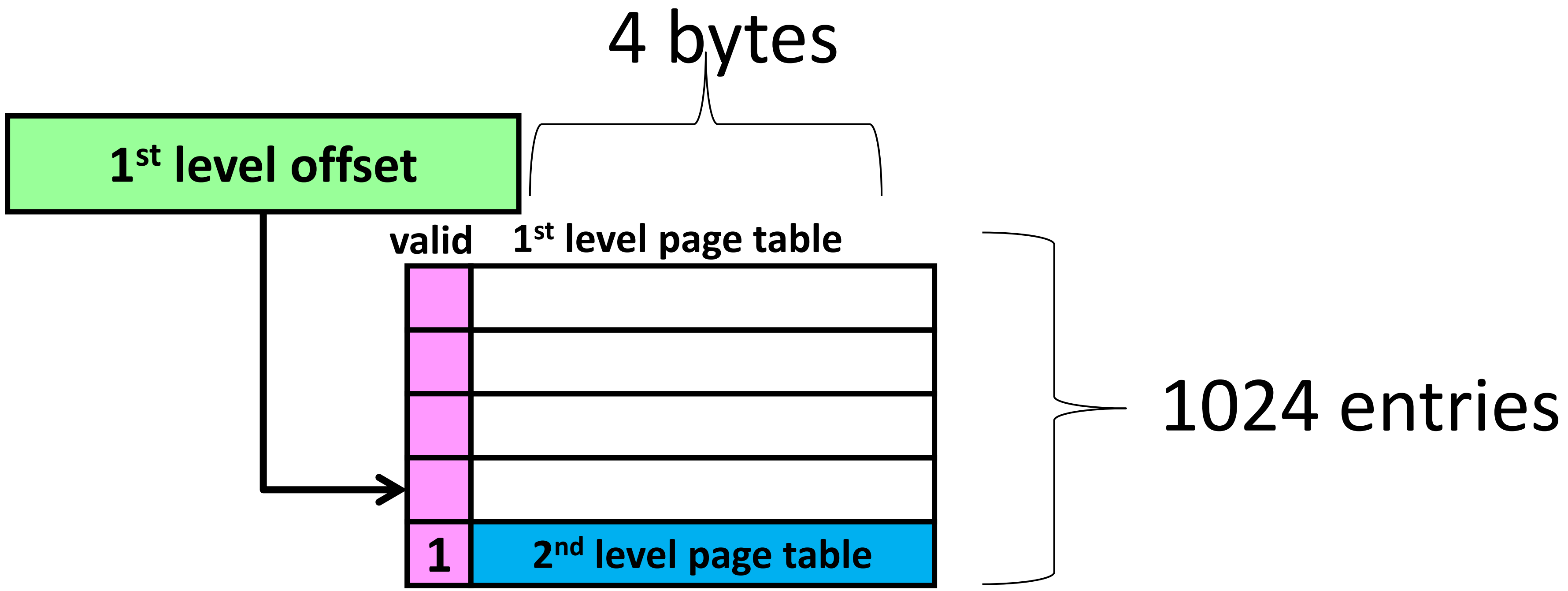
- How many bits in the virtual 1st level offset field? 10
- How many bits in the virtual 2nd level offset field? 10
- How many bits in the page offset? 12
- How many entries in the 1st level page table? 2^{10}=1024



Hierarchical page table – 32bit Intel x86

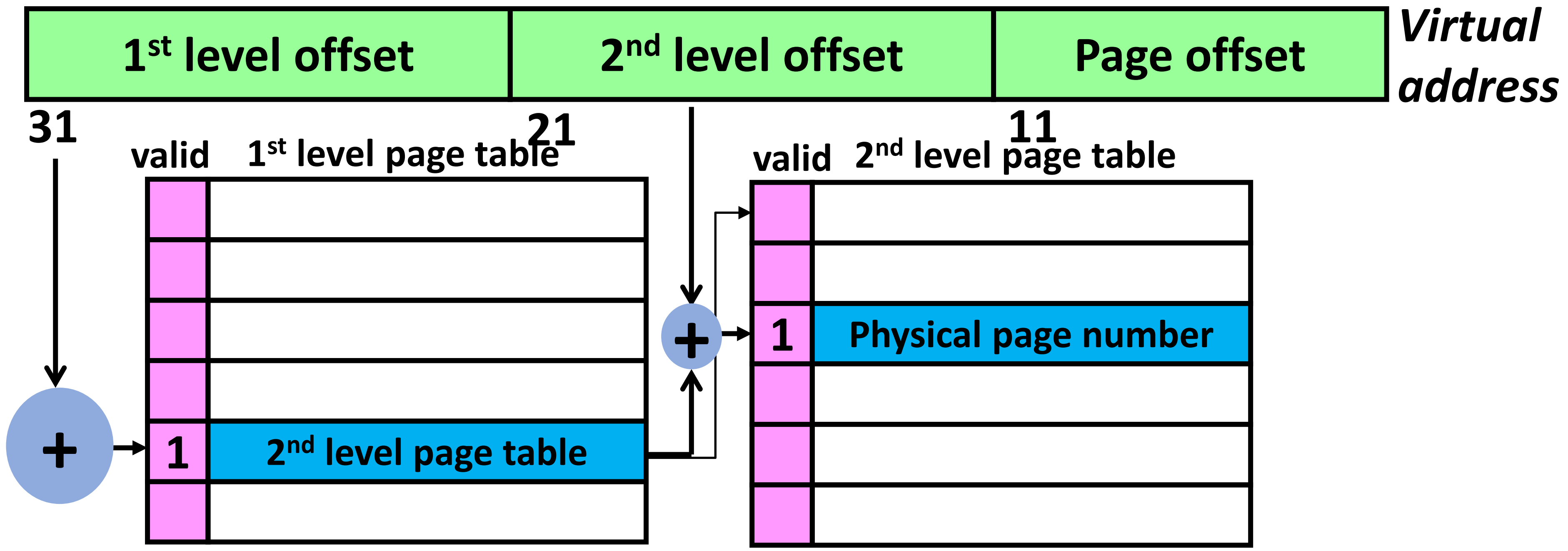


- Let's say physical address size + overhead bits is 4 bytes per entry
- Total size of 1st level page table
  - 4 bytes * 1024 entries = 4 KB

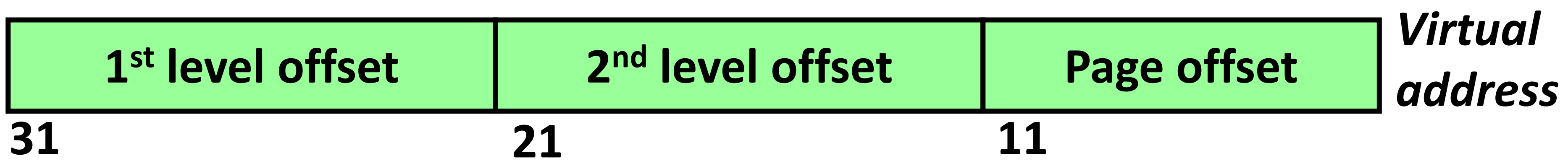


Hierarchical page table

- How many entries in the 2nd level of the page table?
  - $2^{10} = 1024$
- How many bytes for each VPN in a 2nd level table?
  - Let's round up to 4 bytes



Hierarchical page table – 32bit Intel x86



• How many bits in the virtual 1st level offset field?	10
• How many bits in the virtual 2nd level offset field?	10
• How many bits in the page offset?	12
• How many entries in the 1st level page table?	2^{10}=1024
• How many bytes for each entry in the 1st level page table?	4
• How many entries in the 2nd level of the page table?	2^{10}=1024
• How many bytes for each entry in a 2nd level table?	~4
• **What is the total size of the page table?** **(here *n* is number of valid entries in the 1st level page table)**	**4K+n*4K**

Agenda

- Motivation for Multi-level Page Tables
- Example architecture
- **Class Problem: 32bit Intel x86**
- Class Problem: Multi-Level VM Design
- VM Miscellanea

Poll

Class Problem (32 bit x86)

1st 2nd

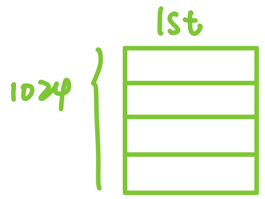
4kB + n·4kB

- What is the least amount of memory that could be used? When would this happen? 4kB. only happen when we start the program and haven't even fetch the first instruction
- What is the most memory that could be used? When would this happen?
- How much memory is used for this memory access pattern:
  0x00000ABC
  0x00000ABD
  0x10000ABC
  0x20000ABC
- How much memory if we used a single-level page table with 4KB pages? Assume entries are rounded to the nearest word (4B)

Class Problem (32 bit x86)

- What is the least amount of memory that could be used? When would this happen?
  - 4KB for 1st level page table. Occurs when no memory has been accessed (before program runs)

- What is the most memory that could be used? When would this happen?
  - 4KB for 1st level page table
    + 1024*4KB for all possible 2nd level page tables
    = 4100KB (which slightly greater than 4096KB) All the second level page table.
  - Occurs when program uses all virtual pages (= 2^{20} pages)



worse case. 1024 active pages. 1024 away from each

But the spacial locality will avoid this.

In the worse case, take slitly more memory than just take 1 level memory.

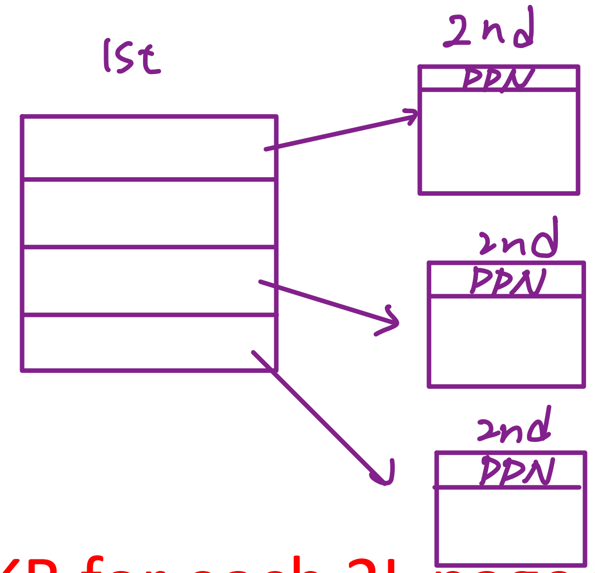
Class Problem (32 bit x86)

if I have 4kB pages page offset is 2^{12} ↙ 12 bits

- How much memory is used for this memory access pattern:

virtual page number page offset

```
0x00000ABC // Page fault
0x00000ABD
0x10000ABC // Page fault
0x20000ABC // Page fault
```



- 4KB for 1st level page table + 3*4KB for each 2L page table = 16 KB

Agenda

- Motivation for Multi-level Page Tables
- Example architecture
- Class Problem: 32bit Intel x86
- **Class Problem: Multi-Level VM Design**
- VM Miscellanea

Class Problem – Multi-level VM

this is virtual address.

Are they mean virtual or physical ?

- Design a two-level virtual memory system of a byte addressable processor with ***24-bit long addresses***. No cache in the system. 2^{18} **256Kbytes of memory** installed, and no additional memory can be added.

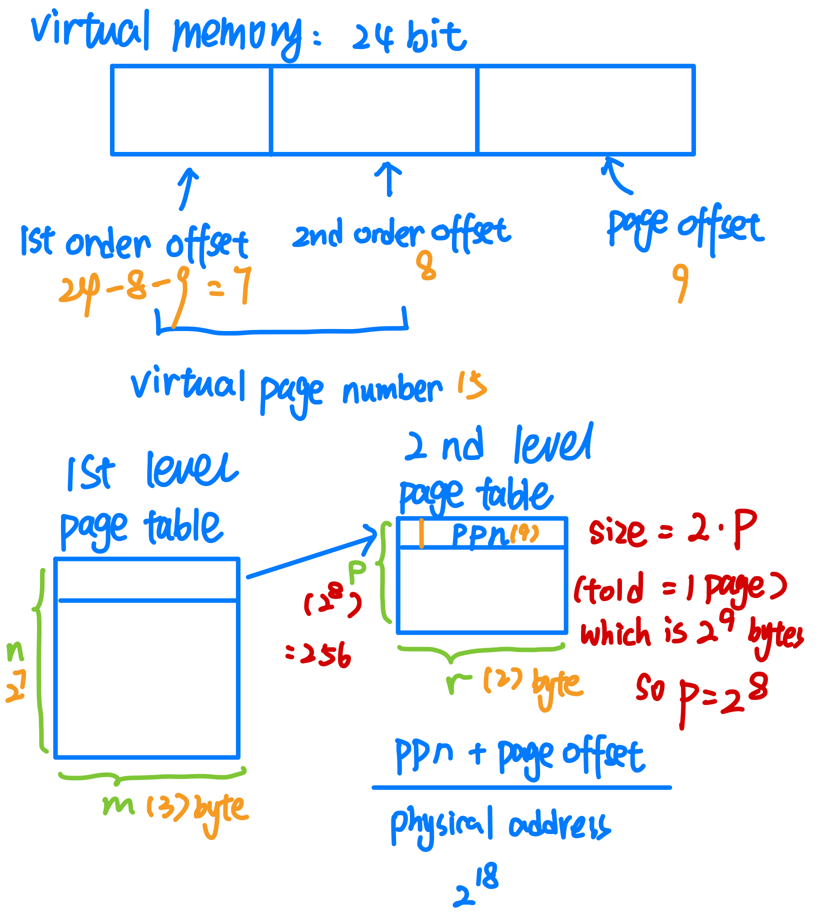
  this is physical address

  We don't install virtual memory.
  - **Virtual memory page: 512 Bytes**. 2^9 Each page table entry must be an integer number of bytes, and must be the smallest size required to fit the physical page number + 1 bit to mark valid-entry
  - We want each second-level page table to fit exactly in one memory page, and 1st level page table entries are 3 bytes each (a memory address pointer to a 2L page table).
- Compute:
  - Number of entries in each 2nd level page table;
  - Number of virtual address bits used to index the 2nd level page table;
  - Number of virtual address bits used to index the 1st level page table;
  - Size of the 1st level page table.

Class Problem – Multi-level VM

- Design a two-level virtual memory system of a byte addressable processor with ***24-bit long addresses***. No cache in the system. **256Kbytes of memory** installed, and no additional memory can be added.
  - **Virtual memory page: 512 Bytes**. Each page table entry must be an integer number of bytes, and must be the smallest size required to fit the physical page number + 1 bit to mark valid-entry
  - We want each second-level page table to fit exactly in one memory page, and 1st level page table entries are 3 bytes each (a memory address pointer to a 2L page table).
- Compute:
  - Number of entries in each 2nd level page table;
  - Number of virtual address bits used to index the 2nd level page table;
  - Number of virtual address bits used to index the 1st level page table;
  - Size of the 1st level page table.

Class Problem – Multi-level VM



- Since the physical address is 2^{18}
  and the virtual memory page = physical memory page $= 2^9$
  So physical page number in 2nd level page table is $2^{18} \div 2^9 = 2^9$
- Also need to plus 1 bit valid-entry.
  So $r = 1 + 9 = 10$.
- Since each table entry must have integer number of bytes.
  So it should have 2 bytes → 16 bits.

- The total number of virtual pages.: 2^{15} total entries for my second level page table.

Class Problem – Multi-level VM

Page Offset: 9 bits (512B page size)

Physical address = 18b (256KB Mem size)

Physical page number = 18b (256KB mem size) – 9b (offset)

Physical page number = 9b	**Page offset = 9b**

2nd level page table entry size: 9b (physical page number) + 1b =~ 2 bytes

2nd level page table **fits exactly in 1 page**

#entries in 2nd level page table is 512 bytes / 2 bytes = 256

#entries in 2nd level page table = 256 ➔Virtual page bits = 8b

Virtual 1st level page bits = 24 – 8 – 9 = 7b

1st level page table size = 2^7 * 3 bytes = 384B

1st level = 7b	**2nd level = 8b**	**Page offset = 9b**

Virtual address = 24b

Agenda

- Motivation for Multi-level Page Tables
- Example architecture
- Class Problem: 32bit Intel x86
- Class Problem: Multi-Level VM Design
- **VM Miscellanea**

Page Replacement Strategies

- Page table indirection enables a fully associative mapping between virtual and physical pages.

- How do we implement LRU in OS?
  - True LRU is expensive, but LRU is a heuristic anyway, so approximating LRU is fine
  - Keep a “***accessed” bit per page***, cleared occasionally by the operating system. Then pick any “unaccessed” page to evict

Other VM Translation Functions

- Page data location
    - Physical memory, disk, uninitialized data
- Access permissions
    - Read only pages for instructions
    - **This is how your system detects segmentation faults**
- Gathering access information
    - Identifying dirty pages by tracking stores
    - Identifying accesses to help determine LRU candidate

Next time

- Speeding up virtual memory